title: coinbase.transfer()

Flashbots allows you to pay validators for your transactions through a smart contract by using block.coinbase.transfer(AMOUNT_TO_TRANSFER). This smart contract function transfers Ethereum from the contract to the address of the validator who proposes a block. The Flashbots builder will treat fees through coinbase transfers in the same way they do normal transaction fees, which is to say that 1 wei of coinbase payments is equivalent to 1 wei paid through transaction fees. This provides significant benefits to Flashbots users: * You can condition payment to the validator on some criteria being met * Related, you can only pay for successful transactions, not failures * You can pay for a transaction from account X with ETH from account Y (see: searcher sponsored transaction repo here)

Here's an example from our open source simple arbitrage bot of how paying through coinbase transfers work:

```solidity function uniswapWeth(uint256 _wethAmountToFirstMarket, uint256 _ethAmountToCoinbase, address[] memory _targets, bytes[] memory _payloads) external onlyExecutor payable { require (_targets.length == _payloads.length); uint256 _wethBalanceBefore = WETH.balanceOf(address(this)); WETH.transfer(_targets[0], _wethAmountToFirstMarket); for (uint256 i = 0; i < _targets.length; i++) { (bool _success, bytes memory _response) = _targets[i].call(_payloads[i]); require(_success); _response; }

uint256 _wethBalanceAfter = WETH.balanceOf(address(this));
require(_wethBalanceAfter > _wethBalanceBefore + _ethAmountToCoinbase);
if (_ethAmountToCoinbase == 0) return;

uint256 _ethBalance = address(this).balance;
if (_ethBalance < _ethAmountToCoinbase) {
    WETH.withdraw(_ethAmountToCoinbase - _ethBalance);
}
block.coinbase.transfer(_ethAmountToCoinbase);

} ```

The above smart contract code will attempt to capitalize on arbitrage opportunities. If it does not make money doing so then the transaction will fail.

For more information on how coinbase transfers are priced see thebundle pricing page.

## Managing payments to coinbase.address when it is a contract

Validators will occasionally have a smart contract listed as their block.coinbase address. This changes the expected behavior of making payments to block.coinbase. Specifically it costs more gas to transfer ETH to block.coinbase if it is a contract than if it is an EOA, and as such many searchers will underestimate their gas consumption and their bundles will fail for validators who use contracts instead.

To handle this edge case searchers can up their gas limit to accommodate the additional payment to validators and call block.coinbase in the following way:

solidity block.coinbase.call{value: _ethAmountToCoinbase}(new bytes(0));

However, searchers should be acutely aware of the risk ofreentrancy attacks, as calling coinbase in this way temporarily gives execution to a third party, and typically payments to coinbase are made after checks for profit. Moreover, searchers should be aware that supporting payments to coinbase addresses that are contracts will cause their gas consumption to go up, and as a result their bundle gas price to go down. This is a tradeoff that should be considered.
```